Re-identifying addresses in LoRaWAN networks

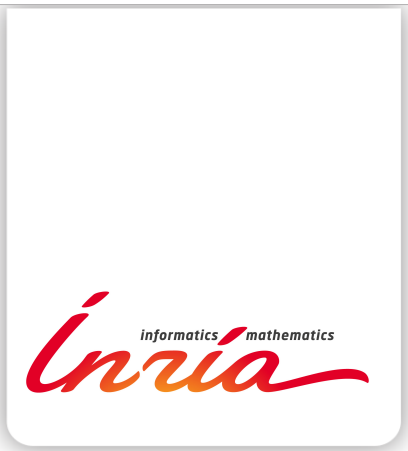
Lucas Toulier Ancian, Mathieu Cunche

Re-identifying addresses in LoRaWAN networks

Lucas Toulier Ancian, Mathieu Cunche

RESEARCH REPORT
N° 9361
August 2020
Project-Teams Privatics

ISSN 0249-6399 ISRN INRIA/RR--9361--FR+ENG



Re-identifying addresses in LoRaWAN networks

Lucas Toulier Ancian, Mathieu Cunche

Project-Teams Privatics

Research Report n° 9361 — August 2020 — 24 pages

Abstract: LoRaWAN is a long range, low energy and low throughput network technology used to provide connectivity to all kind of devices. Encryption ensure the confidentiality of the conveyed data but do not protects metadata, including the `DevAddress` which is the device identifier allocated by the LoRa network. Because of the long range of the radio signals and the open nature of the wireless medium, this metadata can be easily collected and can leak potentially sensitive information about a system communicating through LoRa [11].
In addition to the `DevAddress`, a device is also identified by a `DevEUI`, a globally unique and permanent identifier. As opposed to the `DevAddress`, the `DevEUI` is a static identifier and it can generally be linked to the device manufacturer or the device type. The `DevEUI` is thus a source of additional information that could be combined with the traffic metadata to gain additional knowledge on a device. However, on the wireless link the `DevEUI` is only exposed during the join procedure and is never directly associated with other identifiers, and in particular not with the `DevAddress`.
In this document, we focus on this problem and present a method to link a `DevEUI` to a `DevAddress` and thus to the associated metadata. Our method relies on the time correlation between messages exchanged during the device activation and registration on the LoRa network. The proposed method is tested on two sets of LoRa traces: a real-world dataset and a synthetic one. The corresponding simulation results shows that a significant fraction of the `DevEUI` can be matched to a `DevAddress`. Finally we discuss a number of measures that could be adopted to reduce the efficiency of the presented address linking attack.

Key-words: addresses, identifier, privacy, re-identification, LoRaWAN

RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Ré-identification d'adresses dans les réseaux LoRaWAN

Résumé : LoRaWAN est une technologie réseau à longue portée, faible débit et basse consommation d'énergie utilisée pour fournir une connectivité à toutes sortes d'appareils. Le chiffrement assure la confidentialité des données mais ne protège pas les métadonnées, en particulier la `DevAddress` qui est l'identifiant de l'appareil alloué par le réseau LoRa. A cause de la longue portée des signaux radio et de la nature ouverte du medium sans-fil, ces métadonnées peuvent être aisément collectées et peuvent exposer des informations potentiellement sensibles à propos d'un système [11].

En plus de la `DevAddress`, un appareil est aussi identifié par un `DevEUI`, un identifiant unique. Contrairement au `DevAddress`, le `DevEUI` est statique et il peut être rattaché à un constructeur ou au type de l'appareil. Le `DevEUI` est donc une source d'information qui pourrait être combiné avec d'autres métadonnées. Cependant, sur le canal radio, le `DevEUI` est seulement exposé durant la procédure d'association et n'est jamais directement associé à d'autres identifiant, et en particulier jamais avec le `DevAddress`.

Dans ce document, nous traitons ce problème et présentons une méthode permettant de lier un `DevEUI` à un `DevAddress`, et ainsi aux métadonnées associées. Notre méthode repose sur une corrélation temporelle entre les messages échangés durant la phase d'activation et d'association au réseau LoRa. Cette méthode est testée sur un jeu de traces réel et un jeu de traces synthétique. Les résultats de simulation obtenus montrent qu'une fraction significative des `DevEUI` peuvent être associé à un `DevAddress`. Nous terminons en proposant des mesures qui pourraient être adoptées pour empêcher cette attaque.

Mots-clés : adresse, identifiant, vie privée, ré-identification, LoRaWAN

1 Introduction

As part as the Internet of Things (IoT) a number of low resources devices are currently deployed. To cope with the constraints of those devices, new telecommunication technologies called LPWAN (Low Powered and Wide Area Network) have emerged. One of the prominent LPWAN technology is the *LoRaWAN* protocol [9] whose wireless link is called *LoRa*. LoRaWAN has been largely adopted worldwide and there are millions of device across 133 networks in 143 countries [1] currently using this technology.

In LoRaWAN, messages are first transmitted over a wireless link before being forwarded over the Internet to the destination server. When transiting over this wireless link, messages can easily be recorded by any eavesdropper in range. To cope with associated security issues, LoRaWAN includes several mechanisms that enforce, among others, the confidentiality of the payload [9, Section 4.3.3].

Encryption protects the confidentiality of the payload but the remainder of the packet is left in clear, potentially exposing metadata. Another source of metadata is the temporality of transmission which could reveal the type of the device (e.g. a temperature sensor reporting at the same time every day). The traffic metadata can also reveal information about the device status and its surrounding (e.g. a presence detector will emit only if a presence is detected) [11].

Another key element of metadata exposed in LoRaWAN wireless traffic are the identifiers: the `device address` (`DevAddress`) that serves to identify the device with regard to the base station and the device `EUI` (`DevEUI`) which identifies the device on the LoRaWAN network. Included in every data packet the `DevAddress` can help an eavesdropper to associate wireless activity to entities allowing him to derive information by leveraging the temporal feature of the traffic [11]. On the other hand, the `DevEUI` is a globally unique identifier that can be leveraged to infer other information on the device (manufacturer, type, or the associated application and sensor) that is only exposed during the association process.

By linking a `DevAddress` with a `DevEUI`, an adversary could combine the information brought by each element and thus increase its knowledge on the device, its activities and the associated applications. In LoRaWAN, the link between a `DevEUI` and a `DevAddress` address is only available to owner of the device. An eavesdropper cannot directly link a `DevEUI` to a `DevAddress` as there is no message in which both `DevAddress` and `DevEUI` are included in clear-text.

In this work, we present a technique for identifying the `DevAddress` associated to a `DevEUI`. Our approach is based on temporality of `join request` emitted by a device and the first message emitted by this device following the `join request`. We present a temporal linking method and evaluate it on a real-world dataset comprising 51 000 `join requests`, as well as on a synthetic dataset. Those first results indicates that a re-identification rate of up to 90% could be achieved. Finally we present several counter-measures that could be used to protect devices against this attack.

2 LoRaWAN

LoRaWAN is a media access control (MAC) protocol based on LoRa, a wireless technology for long-range and low-energy transmission. LoRa frames are limited in size (payload of maximum 222 bytes) and the number of frame sent by a device is constrained by a minimum delay between two consecutive frames[1] and a maximum number of frame per unit of time[2]. Together, minimum delay and maximum number of frames define the duty cycle. Local regulations [2] defines the

[1] At least time to send the frame + 2 *time window for receiving frame

[2] For instance `The Things Network` fair policies are of 30s of `uplink messages` per day per node.

maximum duty-cycle a device must respect relative to a regional ISM band (a duty cycle of 1% is enforced in France). LoRa signals are modulated using chirps with various spreading factors and are transmitted over 16 channels (some regions allow more than 16 channels) located in the 433.05-434.79 MHz or 863-870 MHz bands.

2.1 Architecture

A LoRaWAN network is composed of end-devices, gateways, network servers and finally application servers, in a star-of-star topology. End-devices and gateways communicate using the LoRa radio link, while communications between gateways and servers use standard IP connectivity. Each gateway can forward messages to any known network server, leading to a community of LoRaWAN gateways that helps increasing the global coverage. It is possible to add devices around existing gateways with no interaction with the gateways owner, only the network and the application have to be registered to a network participating to the *LoRa-alliance*, the non-profit organization promoting LoRaWAN standard[3].

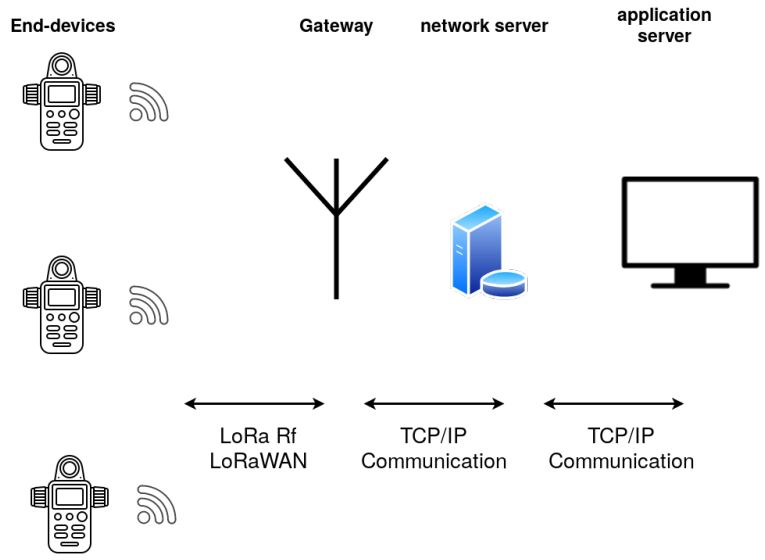


Figure 1: Architecture of a LoRaWAN network.

Messages sent by an end-device are called `uplink messages`, while messages sent by the gateway to end-devices are called `downlink messages`. After their activation, LoRaWAN devices are asynchronous as they send their `uplink messages` whenever they need to. To correctly receive `downlink messages` from a gateway, a device has to be in receiver mode. Gateways can only transmit `downlink messages` when they know the device is in receiver mode.

2.2 End-device

LoRaWAN describes [9] three classes of device (A, B and C) defining their communication features.

A class A device, supports the minimal features. It can send `uplink messages` whenever it needs to. Furthermore, after each `uplink message` transmission, it stays in receive mode during

[3] `https://lora-alliance.org/`

two downlink receive windows, waiting for potential `downlink messages`. Class A devices thus maintain a low energy consumption, by only activating their radio when they are required and at their initiative.

Class B, extends class A by allowing devices to periodically emit an `uplink message` to offer an opportunity for a `downlink message`.

Class C devices, further extends those features by allowing for `downlink message` reception whenever the device is not sending an `uplink message`.

2.3 Frame format

A LoRaWAN frame contains an header and a payload. The header is at least 13 bytes long; it contains the device's `DevAddress` and information about frame options. As the information contained in the header (in particular the `DevAddress`) needs to be available to any receiver, the header is not encrypted. The payload contains the transmitted data. It is encrypted using AES-CTR 128. The maximum payload size depends on the spreading factor, from 51 bytes with SF=12 to 222 bytes with SF=7. A MIC (Message Integrity Code) field is added after the payload; it is computed over the header and the payload using AES-CMAC 128 with the `networkSessionKey` as a key. This MIC is used by the network to validate the integrity of a frame.

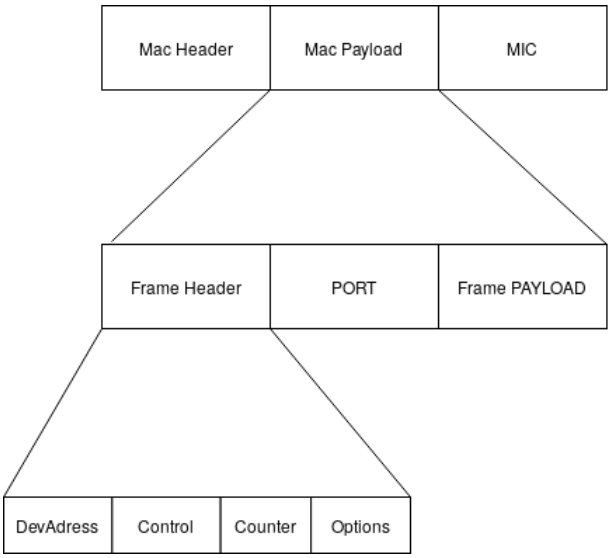


Figure 2: Detail of a LoRaWAN `uplink message`.

2.4 Addressing

A device has a globally unique EUI64 identifier (`DevEUI`) from its constructor, used as a MAC address. The first 3 bytes contain the OUI (Organizationally Unique Identifier) of the constructor registered from IEEE. This `DevEUI` should be unique during the device's life and stored on non-volatile memory.

LoRaWAN added a second temporary identifier, in order to use a same device over different network. This device address (`DevAddress`) is a 32 bits identifier generated by the Network

server during the join procedure. The seven first bits of the `DevAddress` are called the NwkID [4] and corresponds to the 7 first bits of the network identifier (NetID), while the 25 last bits are randomly chosen. The NetID allows device to roam across different network and allow routing of messages to the correct network. NetID are attributed by the Lora-Alliance to each network provider. A `DevAddress` can be used by multiple device at the same time, even devices from the same application, thus it may not be used as a globally unique identifier[5]. To uniquely identify a a device, the network uses the tuple {`DevAddress`, networkSessionKey}. Network servers know for each device the `DevEUI`, the current `DevAddress` and the networkSessionKey. They map a message to the right `DevEUI` and appEUI using this information.

2.5 Personalization and Activation

A device has to be personalized and activated to be able to send `uplink messages`. The personalization part requires to register the device to the network server. This registration consists in configuring the `DevEUI`, the `appEUI` (Application EUI) and the `appKey` (Application key) on the two parties [9, Section 6.2]. The activation part requires one of the two process (OTAA or ABP) to generate the `DevAddress` and the two keys : `nwkSKey` (networkSessionKey) and `appSKey` (applicationSessionKey). This information is only known by the end-device, the network server and the application server. Depending on the process for the activation step, sharing this information is done by a LoRaWAN canal (OTAA) or by a third party channel (ABP).

Over-The-Air-Activation - OTAA With Over-The-Air-Activation, the steps of the activation process, known as "join procedure" are performed directly over the LoRa wireless link through an exchange of messages between the device and the network. The OTAA assumes that the network server already knows the `DevEUI`, the appEUI and the appKey that will be used by the device. Only the appEUI is shared through multiples devices, all the others parameters are specific to a device. Through the OTAA the network will allocated a `DevAddress` to the device and record the association with the `DevEUI`.

The join procedure is in two steps. First, the device send to the network a `join request` including its `DevEUI`, the appEUI and a random nonce called `devNonce`. Then the network replies with a join accept message containing the `DevAddress` allocated by the network and a random nonce called the `appNonce`. This downlink payload is encrypted using the appKey, only the device can decrypt it. Finally, the `devNonce` and `appNonce` are used to compute the `networkSessionKey`, a secret key shared by the device and the network used to encrypt the payload and to verify its integrity through a MIC.

The `DevEUI` is only present in the `join request` while the join accept only contain the `DevAddress`, encrypted. The gateway send the downlink join accept to all listening devices, but only the right device can decrypt it and get the `DevAddress`.

If the OTAA fails, the end-device enter a *waiting* state and will retry after a period of time. An OTAA without need to enter the `waiting` state will last few seconds. [18]

After the OTAA step, the `DevEUI` is never included in any message, and only the `DevAddress` is exposed as part of the header of all `uplink messages`.

OTAA is the recommended process for LoRaWAN activation for maintainability and security [8].

Activation-By-Personalization- ABP With Activation-By-Personalization (ABP), the `DevAddress` and the associated NetworkSessionKey are preconfigured on the device and the network before-

[4] NwkID is renamed AddrPrefix in LoRaWAN 1.1

[5] `https://www.thethingsnetwork.org/docs/lorawan/addressing.html`

hand. A device configured with ABP does not need to go through the join procedure and can start operating (sending and receiving messages) as soon as it is powered on.

While ABP is quicker to activate, this activation can only be done once in a lifetime of a device. A physical access to the device is required before every new activation to update its default parameters.

2.6 Post activation

Once activated a device can start communicating through the network, by either sending `uplink messages` or receiving downlink messages. In a typical scenario, a device will undergo the activation procedure and will then transmit `uplink messages`, and potentially receive downlink messages, over a period of time that can range from weeks to months[14].

2.7 Exposed information

Messages exchanged over the LoRa radio link can be easily collected by an attacker and can expose some information. All uplink and downlink messages contain a payload, encrypted with a state of the art algorithm, thus ensuring the confidentiality of the corresponding data.

Other elements, especially the headers, are not encrypted and can reveal information on the device and its application. The `DevEUI` contains some information about the constructor of the device, while the `DevAddress` contains information about the nwkID.

3 Dataset & first Observations

3.1 Dataset

For this study, we use a dataset of `join requests` and `uplink messages` captured by multiple gateways dispatched around the *La Doua* university campus, at Villeurbanne in France. The dataset contains 51 301 `join requests` and 88 260 `uplink messages` sent through gateways. Data is collected from December, 1st 2019 to May, 31st 2020. From February 25th to March 7th, the gateway was offline and there is thus a visible gap in the dataset.

Each `join request` is identified by its `DevEUI`, while each `uplink message` it identified by its `DevAddress`. An `uplink message` is either *unconfirmed data up* or *confirmed data up*. Their difference lies in the response from the network server. This does not impact the Activation process and the information we use from the data up is identical, thus we regroup the data up types as one named `uplink message`. An `uplink message` contains an id (the `DevAddress`), the time of its first appearance, its last appearance and the number of appearance. We have little information regarding the devices deployed in the area around our gateways. As such we do not know the number of devices or the associated applications.

3.2 Observations

Join requests: During the period covered by our dataset, 751 339 `join requests` were sent. This `join requests` are aggregated based on their `DevEUI` to identify unique devices. We obtain 51 301 `DevEUI` out of this 751 339 `join requests`. Our dataset is composed of the last `join request` for each `DevEUI`. Timestamp of a `join request` mark the last time its `DevEUI` has been seen by the gateway. We identify a device based on its unique `DevEUI` and the time it activated as the timestamp of the last `join request` of the `DevEUI`.

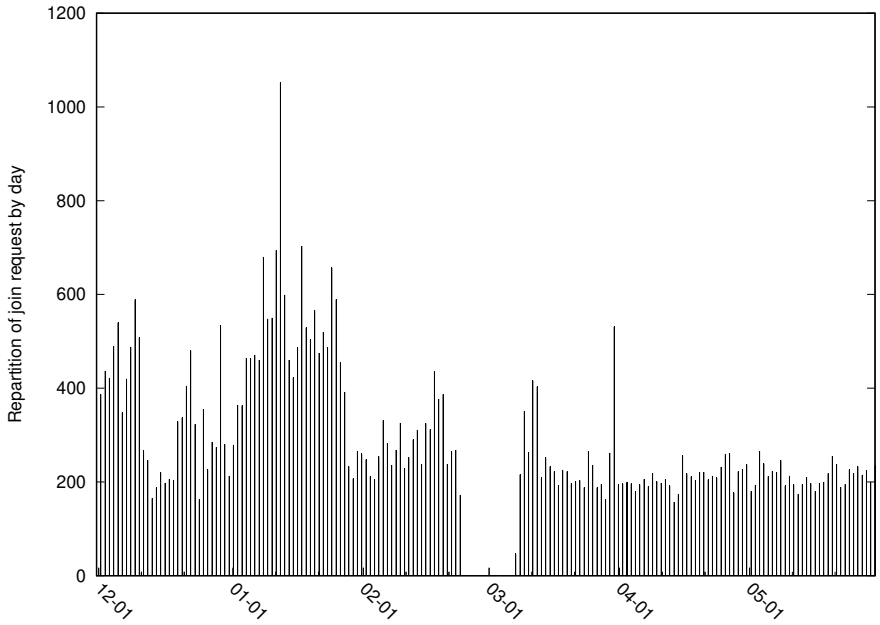


Figure 3: Distribution of join requests each day, between December 1st,2019 and May 31st, 2020

On the figure 3, the distribution shows at least 200 `join requests` per day. The blank period represents a 12-days failure, no data was recorded during this event. During January (between the 31st and the 62nd days of the dataset), each day contains 2 time more `join requests` at least and a peak is visible at January 12th.

`join requests` are sent all day long with a small extra during daytime than during night time. Furthermore, every hour, there is a concentration of `join requests` between the 20th min and the 40th min. On the figure 4, all `join requests` from the 180-day long period are condensed on the same day. The two peaks of 10min (20-30 and 30-40) are visible each hour.

As this is observed each hour, this suggests that an automated process is responsible for this peculiar behavior.

DevEUI Among the 51 301 unique `DevEUI`, 94%sent only one `join request`. Either the devices active during their first try and do not need to re-activate after, or the activation process is not successful but the devices do not try again. The Things Network, a community-oriented LoRaWAN network, recommends to minimize the number of activation [13] to reduce the energy consumption. This expected behavior could explain the very high proportion of `DevEUI` activated only once.

A device's `DevEUI` should be extended from the OUI of the device's constructor or CID (Company ID) based on IEEE Standards Association [4]. The second to last bit from the first octet of a `DevEUI` distinguishes an EUI extended from an OUI or from a CID. As OUI are registered by the IEEE Registration Authority and public, we can link a LoRaWAN device to a

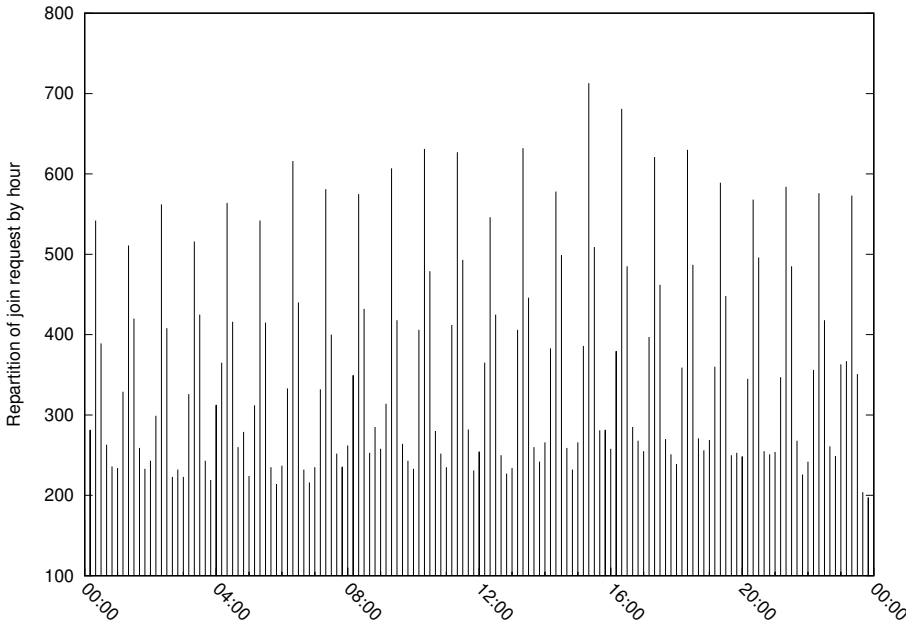


Figure 4: Overall distribution of join request over 24 hours, by bin of 10 mins.

constructor thanks to the OUI. We observe only 15% of `DevEUI` are extended from a OUI. Another part of 33% contains `DevEUI` extended from CID (company ID). The rest (52%) is composed by `DevEUI` with unknown OUI. Either the OUI is not officially registered, or the `DevEUI` is randomly generated.

We focus on `DevEUI` extended from official OUI and unregistered OUI as they contain a group identity. It is expected to find multiple devices from the same manufacturer around the same gateway.

On the figure 6 we represent the percentage of `DevEUI` extended from OUI. There are organizations which appear clearly through their OUI such as : HOMERIDER SYSTEMS (`90-DF-FB`) with 5,78% which represent 2965 `DevEUI`, and IP. Access Limited (`00-02-95`) with 3.98% which represent 2043 `DevEUI`.

In total we detected 521 different OUI (representing 15% of `DevEUI`) and an "OUI_not_registered" group for all the unknown OUI (51% of the `DevEUI`). 323 out of 521 OUI are only represented by one `DevEUI`. Expanding this set to OUI generating less than 5 `DevEUI` (which mean less than 5 unique devices manufactured), we obtain 443 OUI (84%). `DevEUI` of this last set are likely randomly generated and have no real connection with the original OUI's organization.

HOMERIDER SYSTEMS's OUI (`90-DF-FB`) appears in more than 5% of all the `join requests` and seems be an active LoRaWAN device manufacturer. HOMERIDER SYSTEMS corresponds to a former french company now merged in Birdz, an IoT company specialized in water consumption monitoring and probably has devices working around our gateway. Most of the `DevEUI` extended from the HOMERIDER SYSTEMS's OUI are seen only once, possibly meaning the

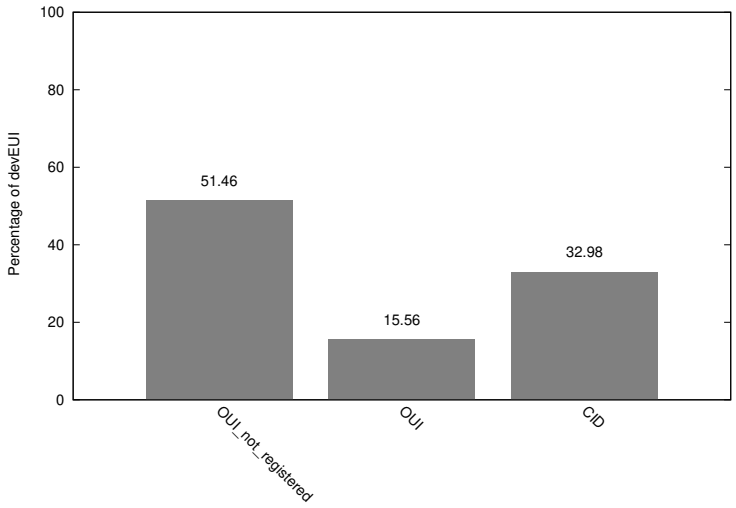


Figure 5: Percentage of devEUI by type of Identifier they are extended

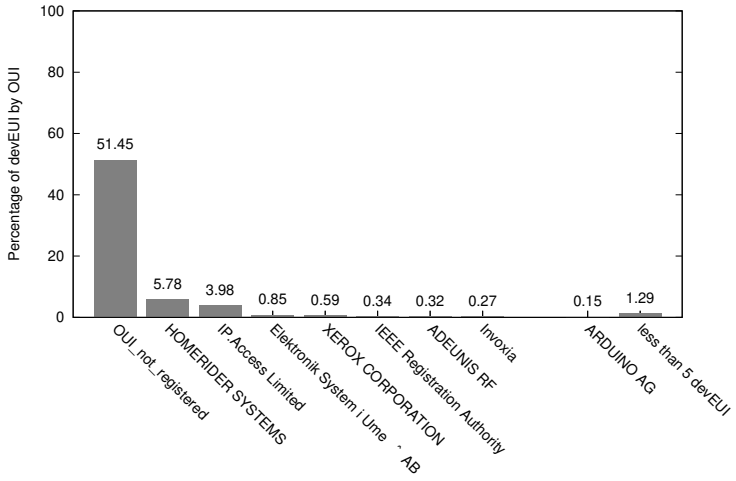


Figure 6: List of most represented OUI

OTAA succeeded on the first try.

ARDUINO AG's OUI extended `DevEUI` are only 79. Inside this small group, 2 `DevEUI` sent

more than 1000 `join requests`, which is not the expected behavior (probably an experiment). On the other side, 53 `DevEUI` sent only one `join request`, these are likely real ARDUINO devices present around our gateway.

UpLink messages During the period covered by our dataset, 2 847 112 `uplink messages` were sent with an average of 32 per `DevAddress`. For each `DevAddress` we kept the time of appearance as the time of the first `uplink message` using this `DevAddress`. The resulting dataset thus contains 88 260 `uplink messages`.

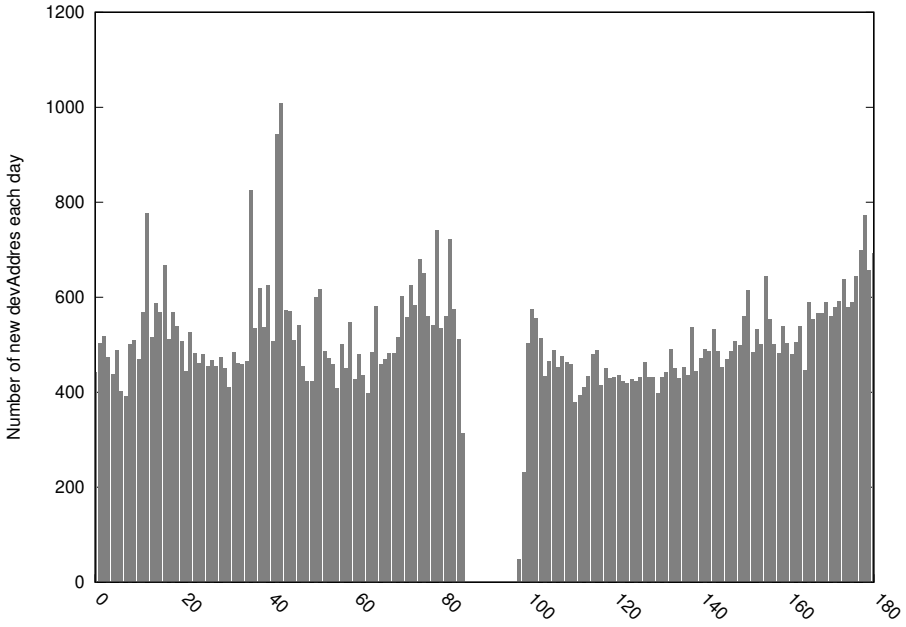


Figure 7: Number of uplink containing new `DevAddress` per day

There is an average of 490 `uplink messages` per day, with extreme short peak of 1 to 2 days. Peaks can be linked to real event such as activation of many device from a new network, or re activation after an outage of the neighbouring gateway and the loss of connection with the network.

DevAddress Among 88 260 unique `DevAddress`, 87% send only one `uplink message` and 96% send 10 or less `uplink messages`. Using the database from The Thing Network website [6], we manage to assign a Network to every `DevAddress`. Among the public networks we detect 88 registered networks.

We found that at least 55% of all `DevAddress` have an non-identifiable nwkID that is either `Network_not_registered` (41.8%), `unassigned` (10.5%) or `Private/experimental nodes`

[6] `https://www.thethingsnetwork.org/docs/lorawan/prefix-assignments.html`

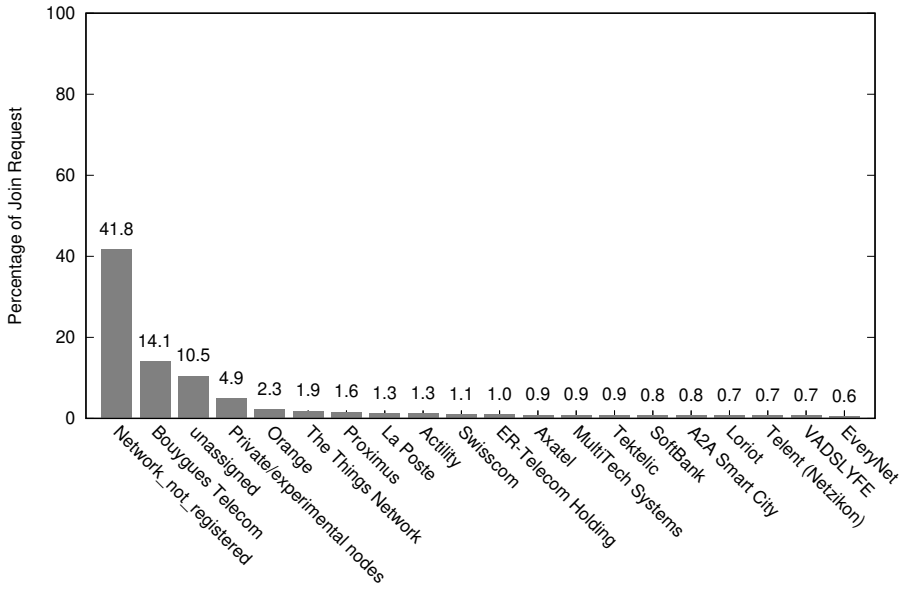


Figure 8: List of most represented `NwkId` and the percentage of number of `uplink messages` for each network

[7] (4.9%). NwkID are the 7 first bits of a `DevAddress`, which lead to identifiable ranges of `DevAddress`. The `Network_not_registered` group refer to `DevAddress` which do not fall into currently registered ranges. This `DevAddress` could come from private or experimental project whit no official network but a dedicated range of addresses. Another possibility come from randomly generated `DevAddress`. Within the total range of nwkID, only 44% are associated to registered networks. Any random `DevAddress` has a fifty-fifty chance (exactly 56%) to contain an unregistered nwkID. `unassigned` represent ranges of `DevAddress` already created but not assigned to a network. We can merge them with the `Network_not_registered` ranges to create a set off ranges identifying an `Unknown` network.

Our gateway is based in France, thus it is expected to detect more French networks such as Bouygues Telecom or La Poste among the most represented networks (8. Bouygues Telecom is a French Internet service providers. Its range represent 0.78% of the possible `DevAddress` values while we detect 14% of the `DevAddress` (12 453 `DevAddress` out of 88 260) which belong to it. It seems there are active devices near our gateways, connected to the Bouygues Telecom LoRaWAN network. Other Internet service providers are also present but with a far smaller number of `DevAddress`, such as Orange (2010 `DevAddress`, 2.3% of all the `DevAddress`) or SwissCom (955 `DevAddress`, 1.59% of all the `DevAddress`).

Selecting the 4 most represented network and the overall, we draw their activation of new `DevAddress` through the period. `Unknown` regroup the previous `Network_not_registered` and

[7] range 00000000 : 03FFFFFF

`unassigned`

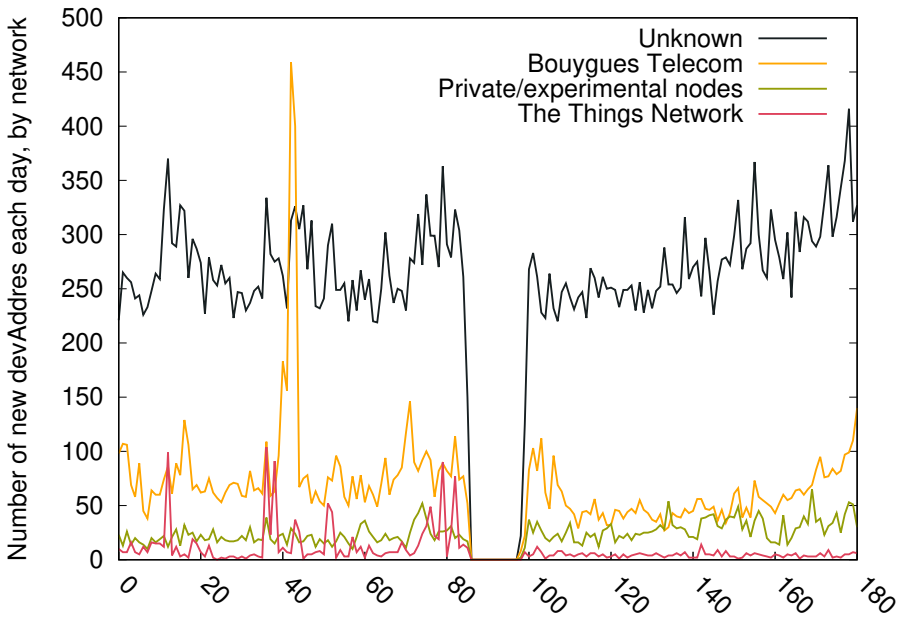


Figure 9: Number of new `DevAddress` per day by network

The Bouygues Telecom network show a clear peak during the 12$^{\text{th}}$ and 13$^{\text{th}}$ January, in parallel of the total peak during this period. During this two days the number of new `DevAddress` is increased by a factor 5. This peak contain 95% of used once `DevAddress`, similar to an `uplink message` test. For reference, the median value of the number of `uplink message` for a Bouygues Telecom's `DevAddress` is 92, which imply a message sent every 2 days in average.

The Things Network is used by an average of 9 new `DevAddress` every day, 6 days show peaks far higher (between 10 to 20 times) than an average day. The highest peak aim to 104 new `DevAddress` the January, 6$^{\text{th}}$ with all of this `DevAddress` sending only one `uplink message`. The 13$^{\text{th}}$ of December, among the 99 new `DevAddress`, only 6 will be really used (between 500 and 900 `uplink messages`) while the 93 other will only be seen once.

Compared to Bouygues Telecom and its 12 453 `DevAddress`, SwissCom has less than ten times less `DevAddress` (955), but most importantly the median number of `uplink message` for a SwissCom's `DevAddress` is only 1.6, which imply at least half of its devices send only one `uplink message`. The SwissCom network does not seems to have devices sending daily `uplink messages` as to what is expected from LoRaWAN devices. The SwissCom network may be operational but it is not actively used.

Global observations We draw the curves representing the number of new `DevEUI` and `DevAddress` per day on the same graph to compare their evolution. As every `join request` should lead to the activation of the device, which then would send an `uplink message`, we expect to have at least

more new `DevAddress` than new `DevEUI` every day. The difference could be devices activated by personalizing.

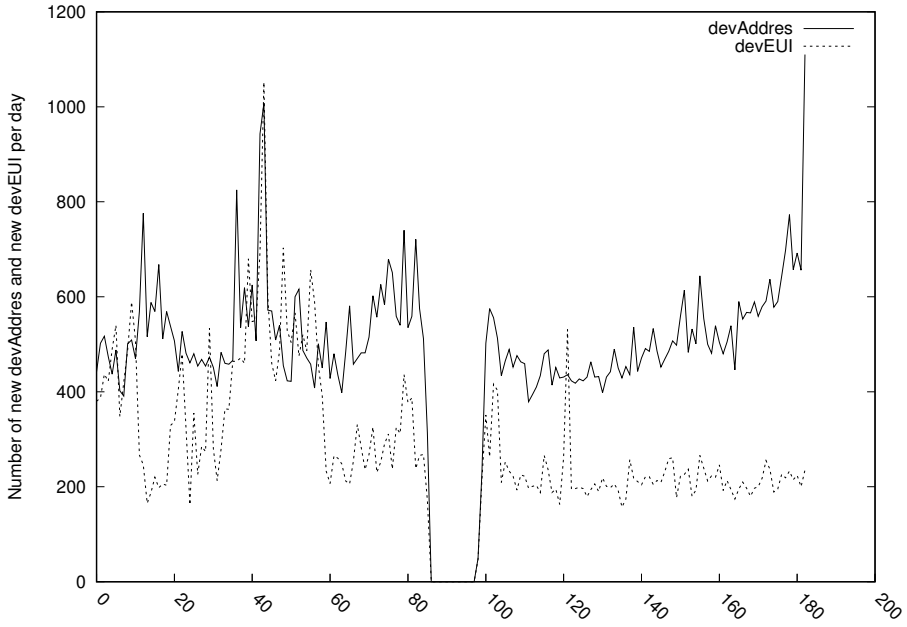


Figure 10: Comparaison of number of new `DevAddress` and new devEUI per day

There is no clear correlation between the curves from the figure 10. The difference of 40% between new `DevAddress` and new `DevEUI` imply that only 60% of the new `DevAddress` are generated during an OTAA. New `DevAddress` can also be generated via Activation By Personalization while devices can display their `DevEUI` during the OTAA but neither send any `uplink message` using their newly attributed `DevAddress` afterward.

Locals matching patterns exist, such as the January 12^{th} to 13^{th} peak. We identify the Bouygues Telecom event during this two days, when the peak of new `DevAddress` is clearly visible on the `DevEUI` curve. All those new `DevAddress` seems to have been generated during the OTAA. The matching peak between the two curves during the same day imply a delay between the OTAA and the first `uplink message` from the new `DevAddress` inferior to a day. On this opposite, the March 31^{st}, more new `DevEUI` were detected than new `DevAddress`. There is no delayed peak from the `DevAddress` curve, which imply most of the new devices were not following the normal use case : either the Activation process failed, or the device received a `DevAddress` in a join-accept message but neither sent an `uplink message` with its `DevAddress`.

4 Re-identifying Dev addresses

In LoRaWAN, a device is identified by two identifiers, the `DevAddress` and the `DevEUI`. The association between those two identifiers is only available to the application and the network.

An external observer monitoring the radio link will not be able to directly link a `DevAddress` to a `DevEUI` since no message contains both elements in clear.

In this section we present a method to re-identify LoRaWAN addresses, i.e. find the `DevAddress` associated to a `DevEUI`. This attack is based on the timing of messages transmitted during and shortly after the activation process. This attack leverages the fact that new `DevAddress` are allocated for each activation. Therefore, a never seen before `DevAddress` appearing on the network can be correlated to a recent activation in which a `DevEUI` has been exposed.

4.1 Hypothesis and attacker model

We make the assumption that a `DevAddress` is unique to a device during our experiment. We assume that a device does not loses its connection and keeps the same `DevAddress`. If a device loses the connection with the network it will try again the OTAA and will be given a new `DevAddress`.

As described in Section 2.6 a devices needs to be activated before transferring data. The LoRaWAN specification recommends the OTAA process for the activation part [8]. We expect each device to send a `join request`, as part of their OTAA, followed by multiples `uplink message`.

Attacker model The goal of the attacker is to associate a `DevAddress` to each `DevEUI` active on the network. We assume that the attacker is able to monitor LoRaWAN traffic on the LoRa radio link, either by monitoring the wireless signals with a dedicated receiver or by having control of a gateway. This can be achieved by controlling or simply setting up a LoRaWAN gateway that cost around 300€[8] or by compiling free code on many small computer costing 100€, such as "SX1308 Raspberry Pi Zero LoRa Gateway Board"[9]. Furthermore, we assume that the attacker is passive, i.e. it only collects messages and never injects messages nor disrupts the communications.

As a result, the attacker has access to header and metadata of frames, and payload of cleartext frames, but cannot decrypt the payload of encrypted frames.

4.2 Re-identification process

In this section we present a method to reidentify `DevEUI`. This method is based on the matching of a `join request`, that includes a `DevEUI`, with an `uplink message`, that includes a `DevAddress`. Following the assumption that a device activate in order to send data, and so will send an `uplink message` shortly after its activation, our approach relies on the timing of those messages to find the correct match.

Our approach works as follows: for each `DevEUI` found in a `join request` we find the first new `DevAddress` appearing after the `join request`; the corresponding `DevAddress` is then associated to the `DevEUI`.

We keep our `join request` list ordered by time and we start to match the oldest `join request` first. To reduce the possibility of incorrect matches, we limit the distance between the two messages to be lower than a threshold `maxTimeOffset`. This means that, for a `join request` sent a time t, only the `uplink message` that are observed before t+`maxTimeOffset` will be considered for matching.

A `DevAddress` is removed of the list of candidates once it has been matched with a `DevEUI`. As such, a `DevAddress` can only be associated once.

[8] `https://www.thethingsnetwork.org/the-things-products#the-things-product-buy`

[9] `https://www.tindie.com/products/will123321/sx1308-raspberry-pi-zero-lora-gateway-board/`

This approach is implemented by the algorithm 1. It takes as input a list of `join request` (objects containing {`DevEUI`, datetime}) and a list of `uplink message` (objects containing {`DevAddress`, datetime}, both ordered by time. Following an iterative approach the algorithm goes through all `join requests`, and for each `join request` goes through all `uplink messages` and select the first one appearing after the `join request` and before `maxTimeOffset`; the corresponding `DevAddress` is then associated to the `join request`'s `DevEUI` in the map M. The corresponding `uplink message` is then removed from the `uplink message` list.

Algorithm 1 Re-identification Algorithm

Input : joinRequestList *: list of joinRequest*, and
upLinkList *: list of upLink ordered by date*
Output: M[devEUI]=devAddress *: for each devEUI, map the corresponding devAddress*

```
for all joinRequest in joinRequestList do
  M[joinRequest.devEUI] ← ∅
  for all upLink in upLinkList do
    if (upLink.time > joinRequest.time) AND (upLink.time < joinRequest.time +
    maxTimeOffset) then
5:    M[joinRequest.devEUI] ← UpLink.devAddress
    end if
  end for
  upLinkList.popOut(M[joinRequest.devEUI])
end for
```

5 Experimental evaluation

We evaluate our algorithm on two datasets. The first dataset is the one presented in Section 3.1 and corresponds to real world capture of LoRaWAN traffic. However, for this dataset, we do not have access to the real association between `DevAddress` and `DevEUI`; we thus lack the groundtruth to fully evaluate the performances of our approach. Therefore, we rely on a synthetic dataset, which contains a similar number of `join request` and `uplink messages` as the real dataset.

The synthetic dataset is composed of 50 000 `join requests` distributed uniformly over a period of 6 month. Each `join request` is associated to a unique `DevEUI`. For each `join request`, we generate an `uplink message` with a delay following the Poisson distribution with $\lambda = 160$, the average delay between a `join request` and the next `uplink message` observed in the real dataset.

For each of our dataset, we execute our algorithm. The threshold `maxTimeOffset` follow this logarithmic scales, in seconds : [2, 5, 11, 26, 61, 138, 316, 719, 1637, 3727, 8483, 19306, 86400, 172800].

Applying our method to the real-world dataset provide an indication of the number of `DevEUI` that can be reidentified and allows us to identify possibles links between networks (from the `DevAddress`) and devices manufacturers (from the `DevEUI`).

Results obtained with the synthetic dataset provides an estimation of the performances of the proposed approach. In particular, it allows to compute the number of true/false positives/negatives and the corresponding metrics. After the execution of our algorithm for each value of `maxTimeOffset`, we define the following quantities:

- **True positive (TP)**: the number of `DevEUI` matched to their corresponding `DevAddress`

- **False positive (FP)**: the number of `DevEUI` matched to a wrong `DevAddress`
- **False Negative (FN)**: the number of `DevEUI` not to `DevAddress` but which should have been

from which are derived the following metrics:

- **Positive Predicted Value (PPV)**: $\frac{TP}{TP+FP}$
- **False Discovery Rate (FDR)**: $\frac{FP}{TP+FP}$
- **True Positive Rate (TPR)**: $\frac{TP}{TP+FN}$

5.1 Results

5.1.1 Performances on the synthetic dataset

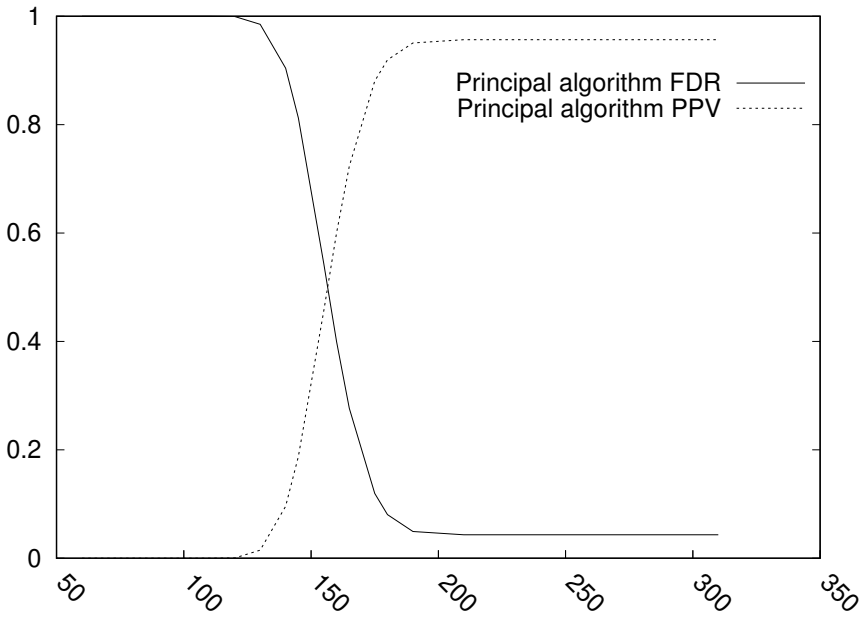


Figure 11: Variation of different metrics as a function of the `maxTimeOffset` applied the generated dataset.

The matching algorithm was applied to the synthetic dataset, and using its ground-truth we computer the corresponding performances. The figure 11 show the variation of the Positive Predicted Value (PPV) and the False Discovery Rate (FDR) as a function of `maxTimeOffset`. Overall the performances increases as the `maxTimeOffset` increases; and in particular in the region around `maxTimeOffset` $= \lambda = 160$, where the PPV grows up from 0 to more than 0.5,

and DFR goes down from 1 to less than 0.5. This result is expected given the way this dataset was generated: the distance between the `join request` and the `uplink message` is centered around $\lambda = 160$ seconds.

After the value `maxTimeOffset` = 200, 94% of the `DevEUI` are matched, with 95% of this matches correct (see Figure 12).

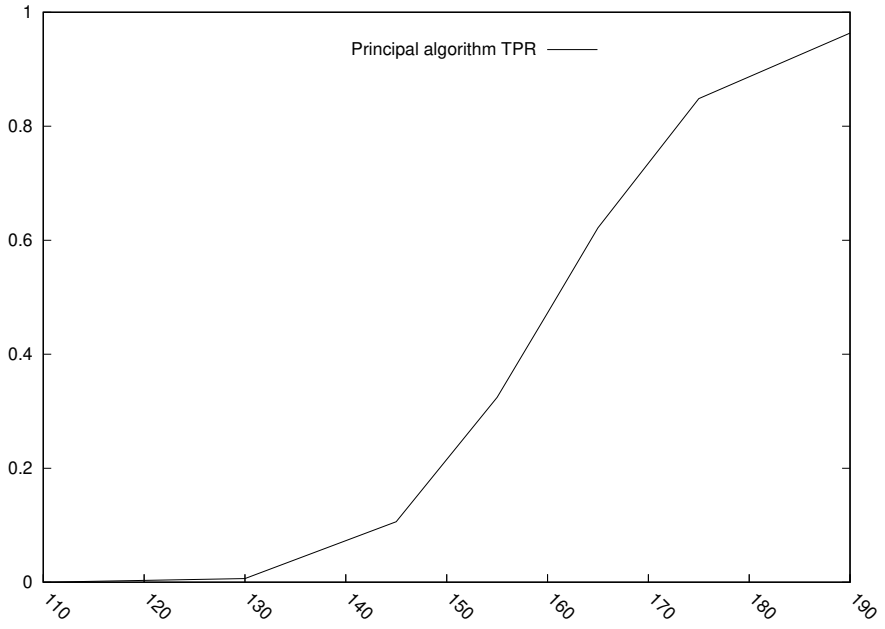


Figure 12: Variation of TPR as a function of the `maxTimeOffset` applied the generated dataset.

5.1.2 Application to the real-world dataset

As the real-world dataset does not include ground-truth, it was not possible to directly evaluate the efficiency of the algorithm. The algorithm was applied to this dataset and a number of re-identification were obtained. Figure 13 presents the fraction of re-identified `DevEUI` as as function of `maxTimeOffset` in the real-world dataset. The fraction of `DevEUI` increases as `maxTimeOffset` increases, but without reaching 100%. For a `maxTimeOffset` of 172 800 seconds (= 2 days), there are still 2% of `DevEUI` that have not been matched. Apart from an error of our re-identification, this could be explained by a failure during the activation or by the lack of `uplink messages` after the `join request`. With a value of `maxTimeOffset`=316, 60% of the `DevEUI` are matched with a `DevAddress`.

Observation on device deployment Leveraging the results of real-world dataset we are able to make some observation such as the correlation between the nwkID and the OUI and peak of activation in a network.

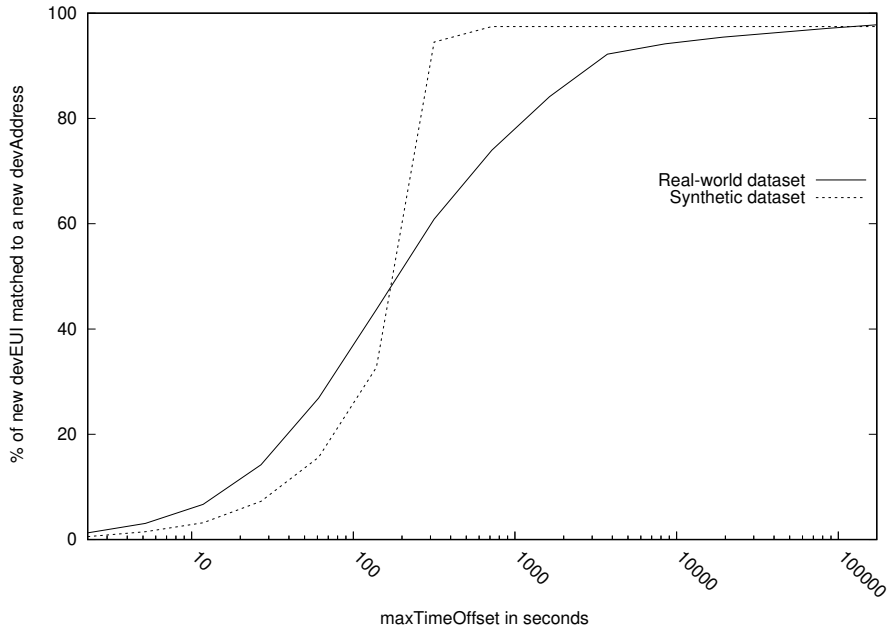


Figure 13: Fraction of re-identified devEUI as a function of the `maxTimeOffset` applied on the realworld dataset and the synthetic dataset.

For each matched pair of `DevEUI`/ `DevAddress`, the `DevAddress` provides the nwkID while the `DevEUI` provides the OUI. Thus, the re-identification allows us to analyze the correlation between nwkID and OUI.

Figure 14 show the distribution of OUI found in the The Things Network network. A majority (80%) of `DevEUI` have a non-registered OUI, and only a small fraction of `DevEUI` are associated to registered OUI corresponding to companies such as Homerider (6.4%) and Arduino AG (3.3 %).

The reidentification result allowed us to spot a peculiar activity on the Bouygues Telecom network between January 11th and 12th. During this period, a peak of device activation was observed on the Bouygues Telecom network as seen on Figure 15. The corresponding `DevEUI` were associated to "Elektronik System i Umeå AB" and the "OUI_not_registered" group. This peak is momentary and the new `DevAddress` are seen only once. This suggest new devices were tested but not deployed.

We observe `DevEUI` extended from the "Elektronik System i Umeå AB" OUI do not represent an important part of the Bouygues Telecom network. They appear mainly during the January peak and May. This event can be interpreted as an activation on the Bouygues Telecom network of devices associated to Elektronik System i Umeå AB. It is difficult to estimate the number of real devices activated but at least one hundred `DevEUI` extended from Elektronik System i Umeå AB were seen.

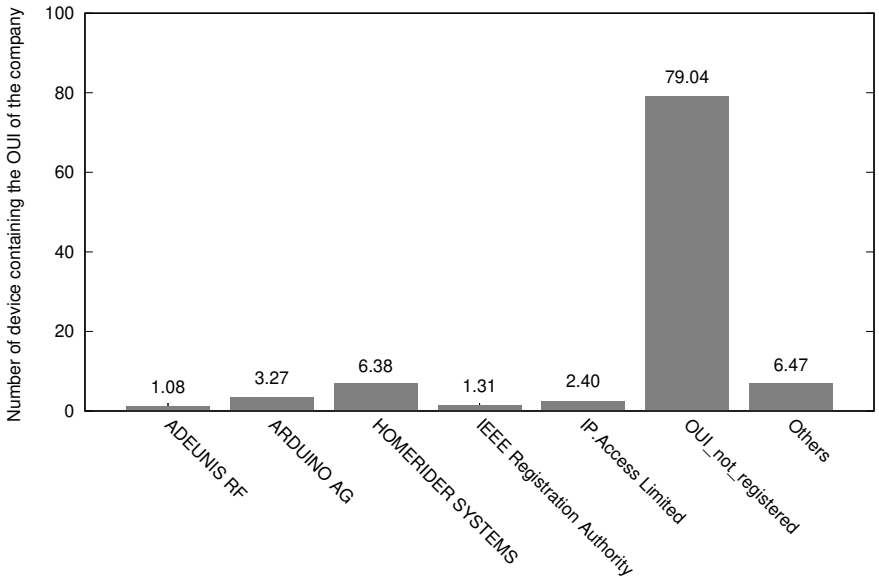


Figure 14: Number of devEUI by company matched to `DevAddress` associated to The Things Network.

6 Counter-measures

This section presents several counter-measures that can be used to reduce the efficiency of the address reidentification attack presented above.

Delaying the first `uplink message` As the attack relies on timing, introducing some perturbation in this timing can increase the difficulty of the attack. More specifically, a random delay can be introduced between the `join request` and the first `uplink message`.

Introduction of delay will have a minimal impact on the application, as it will only impact the device just after activation by introducing a delay in its activation. The larger the delay, the more difficult it will be to reidentify addresses.

Shared DevAddress Another solution to mitigate observation that can be done on a device is to share a same `DevAddress` between multiple devices, thus creating an anonymity set [10]. As multiple devices are using the same `DevAddress`, an attacker would not be able to associate traffic to a specific device but only to a group of device.

Having multiple devices behind the same `DevAddress` appear to be compatible with the specifications. First, the `DevAddress` is not supposed to be uniquely allocated to a device; then in the network, the identification of a device does not only rely on the `DevAddress`, but also on the NwkSessionKey. Therefore, as long as each device in the set uses a different NwkSessionKey,

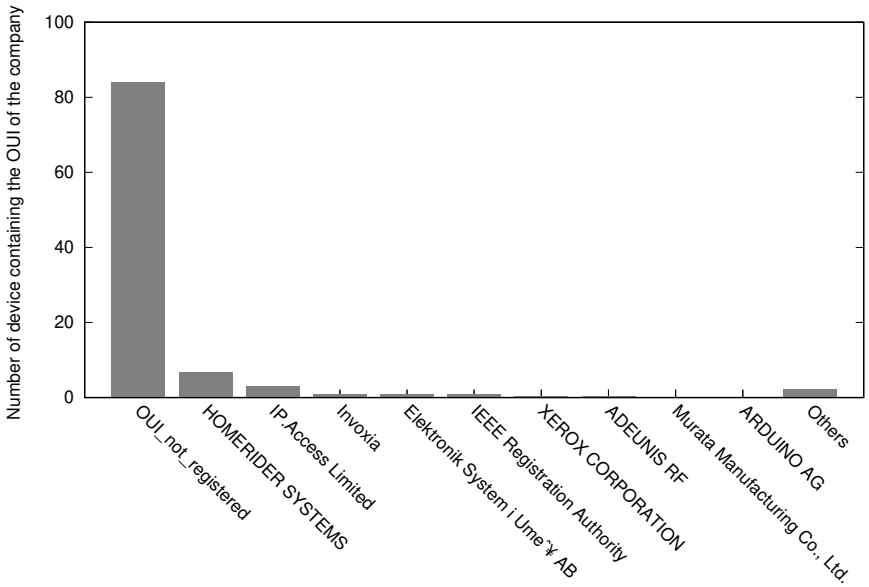


Figure 15: Number of devEUI by company matched to `DevAddress` associated to by Bouygues Telecom network.

the network should be able to associate each message to the correct device.

DevAddress rotation Changing the `DevAddress` of a device without going through a `join request` can hamper the the re-identification attack.

A first way to implement this address change would be to let the device and the network server select a new address through the existing secured channel. The address rotation could be initiated by the device or by the server. To this aim, it will be necessary to define new message types and a new procedure to implement this address rotation functionality.

A second option for `DevAddress` rotation is to use the concept of *resolvable addresses* as used in Bluetooth[16, Vol 6, Part B, Section 1.3.2.2]. Based on a secret key shared by the server and the device, the device can generate addresses that would appear random to an external observer but not to the server that will be able to recognize them by *resolving* them using the shared key.

With each of those solutions, the device will be able to use `DevAddress` that are unlinkable with the `DevEUI` based on the join procedure.

7 Related work

Since its release, LoRaWAN has been the subject of a number of security studies. A general analysis of security features of LoRaWAN is presented in [7]. In there paper [3], Aras et al. present a number of attack scenarios including jamming, replay and wormhole attacks, as well

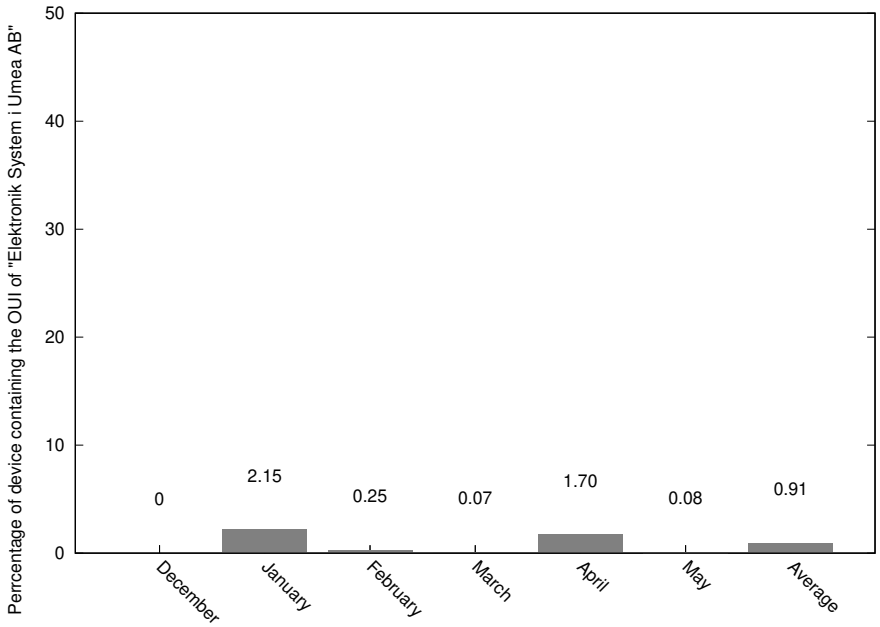


Figure 16: Number of devEUI from Elektronik System i Umeå AB matched to `DevAddress` generated by Bouygues Telecom by month.

as key compromise via physical access. In [17], Tomasin et al. identified a poor management of random number in the join procedure, which could lead to denial of service triggered by replay attacks.

As opposed to security, privacy aspects of LoRaWAN has received little attention. Features of wireless traffic can be leveraged to infer the activities of devices operating on a network. This problem is addressed by Leu et al. [11] in the more general context of LPWAN; they introduce a solution to obfuscate temporal patterns by introducing random delays and injecting dummy traffic.

Identification of wireless device can be done at the physical layer by leveraging features of the raw wireless signal. Physical fingerprinting as been applied to LoRa by Robyns et al. [15]; they demonstrated how a device can be singled-out using machine learning applied to signal collected through software-defined radio.

Timing information and frame content has already been exploited to link device identifiers. Matte et. al showed [12] that timing of 802.11 frames could be leveraged to link the consecutive random addresses used by a device. Pseudonyms belonging to the same device have been linked based on the body of the frame, from the physical [6] layer to the application layer [19, 5].

8 Conclusion

As demonstrated in this paper, the `DevEUI` and the `DevAddress` of a LoRa device can be linked. This attack relies on the join procedure of LoRa protocol and leverage the fact that a `join request` is shortly followed by an `uplink message`. According to our simulation, up to 90% of devices could have their addresses successfully linked. Note that those numbers have been obtained using a naive model for LoRaWAN traffic. As future work, we plan on using a more refined model for generating the synthetic dataset. Furthermore, those results need to be confirmed on a real dataset where the ground truth is readily available.

The information revealed by this attack can be exploited by the attacker to gain additional knowledge on an application running on LoRaWAN. Depending on the nature of the application, this information can threaten the privacy of individuals or could expose sensitive information about a company.

The LoRaWAN specification includes measures to ensure the confidentiality of the data transmitted over the network. However, as of today, privacy threats resulting from side channel leakages are not covered. Some of the countermeasures presented in this paper could be deployed independently from the protocols, but other would require a modification of the specifications.

References

[1] LA_2019_annualreport_external_interactive.pdf.

[2] lorawan_regional_parameters_v1.0.3reva_0.pdf.

[3] Emekcan Aras, Gowri Sankar Ramachandran, Piers Lawrence, and Danny Hughes. Exploring the security vulnerabilities of lora. In *2017 3rd IEEE International Conference on Cybernetics (CYBCONF)*, pages 1–6. IEEE, 2017.

[4] IEEE Registration Authority. eui.pdf.

[5] Johannes K Becker, David Li, and David Starobinski. Tracking anonymized bluetooth devices. *Proceedings on Privacy Enhancing Technologies*, 2019(3):50–65, 2019.

[6] B. Bloessl, C. Sommer, F. Dressier, and D. Eckhoff. The scrambler attack: A robust physical layer attack on location privacy in vehicular networks. In *2015 International Conference on Computing, Networking and Communications (ICNC)*, pages 395–400, 2015.

[7] Ismail Butun, Nuno Pereira, and Mikael Gidlund. Analysis of lorawan v1.1 security: Research paper. In *Proceedings of the 4th ACM MobiHoc Workshop on Experiences with the Design and Implementation of Smart Objects*, SMARTOBJECTS '18, New York, NY, USA, 2018. Association for Computing Machinery.

[8] LoRa Alliance Technical Committee. la_faq_security_0220_v1.2_0.pdf.

[9] LoRa Alliance Technical Committee. lorawan1.0.3.pdf.

[10] Claudia Diaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards measuring anonymity. In *International Workshop on Privacy Enhancing Technologies*, pages 54–68. Springer, 2002.

[11] Patrick Leu, Ivan Puddu, Aanjhan Ranganathan, and Srdjan Čapkun. I Send, Therefore I Leak: Information Leakage in Low-Power Wide Area Networks. In *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks - WiSec '18*, pages 23–33, Stockholm, Sweden, 2018. ACM Press.

[12] Célestin Matte, Mathieu Cunche, Franck Rousseau, and Mathy Vanhoef. Defeating mac address randomization through timing attacks. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, WiSec '16, page 15–20, New York, NY, USA, 2016. Association for Computing Machinery.

[13] The Things Network. Best practices.

[14] Tommaso Polonelli, Davide Brunelli, Andrea Bartolini, and Luca Benini. A LoRaWAN wireless sensor network for data center temperature monitoring. In Sergio Saponara and Alessandro De Gloria, editors, *Applications in Electronics Pervading Industry, Environment and Society*, volume 573, pages 169–177. Springer International Publishing.

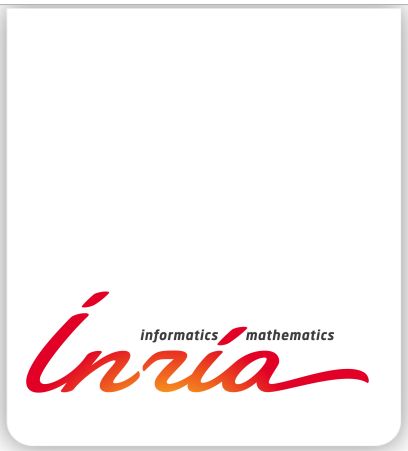
[15] Pieter Robyns, Eduard Marin, Wim Lamotte, Peter Quax, Dave Singelée, and Bart Preneel. Physical-layer fingerprinting of LoRa devices using supervised and zero-shot learning. In *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks - WiSec '17*, pages 58–63, Boston, Massachusetts, 2017. ACM Press.

[16] Bluetooth SIG. *Bluetooth Core Specification v5.1.* 2019.

[17] Stefano Tomasin, Simone Zulian, and Lorenzo Vangelista. Security analysis of LoRaWAN join procedure for internet of things networks. In *2017 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 1–6.

[18] Joel Toussaint, Nancy El Rachkidy, and Alexandre Guitton. Performance analysis of the on-the-air activation in LoRaWAN. In *2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 1–7. IEEE.

[19] Mathy Vanhoef, Célestin Matte, Mathieu Cunche, Leonardo S. Cardoso, and Frank Piessens. Why mac address randomization is not enough: An analysis of wi-fi network discovery mechanisms. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '16, page 413–424, New York, NY, USA, 2016. Association for Computing Machinery.



RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399